|  |  |  |
| --- | --- | --- |
| 1  2  3  4 | # Data Source: <https://archive.ics.uci.edu/ml/machine-learning-databases/car/>    install.packages("randomForest")  library(randomForest) | |
| 1  2  3  4  5  6  7  8 | # Load the dataset and explore  data1 <- read.csv(file.choose(), header = TRUE)    head(data1)    str(data1)    summary(data1) |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | > head(data1)    BuyingPrice Maintenance NumDoors NumPersons BootSpace Safety Condition  1       vhigh       vhigh        2          2     small    low     unacc  2       vhigh       vhigh        2          2     small    med     unacc  3       vhigh       vhigh        2          2     small   high     unacc  4       vhigh       vhigh        2          2       med    low     unacc  5       vhigh       vhigh        2          2       med    med     unacc  6       vhigh       vhigh        2          2       med   high     unacc  > str(data1)  'data.frame':   1728 obs. of  7 variables:   $ BuyingPrice: Factor w/ 4 levels "high","low","med",..: 4 4 4 4 4 4 4 4 4 4 ...   $ Maintenance: Factor w/ 4 levels "high","low","med",..: 4 4 4 4 4 4 4 4 4 4 ...   $ NumDoors   : Factor w/ 4 levels "2","3","4","5more": 1 1 1 1 1 1 1 1 1 1 ...   $ NumPersons : Factor w/ 3 levels "2","4","more": 1 1 1 1 1 1 1 1 1 2 ...   $ BootSpace  : Factor w/ 3 levels "big","med","small": 3 3 3 2 2 2 1 1 1 3 ...   $ Safety     : Factor w/ 3 levels "high","low","med": 2 3 1 2 3 1 2 3 1 2 ...   $ Condition  : Factor w/ 4 levels "acc","good","unacc",..: 3 3 3 3 3 3 3 3 3 3 ...  > summary(data1)   BuyingPrice Maintenance  NumDoors   NumPersons BootSpace    Safety    Condition   high :432   high :432   2    :432   2   :576   big  :576   high:576   acc  : 384   low  :432   low  :432   3    :432   4   :576   med  :576   low :576   good :  69   med  :432   med  :432   4    :432   more:576   small:576   med :576   unacc:1210   vhigh:432   vhigh:432   5more:432                                     vgood:  65 |

Now, we will split the dataset into train and validation set in the ratio 70:30. We can also create a test dataset, but for the time being we will just keep train and validation set.

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8 | # Split into Train and Validation sets  # Training Set : Validation Set = 70 : 30 (random)  set.seed(100)  train <- sample(nrow(data1), 0.7\*nrow(data1), replace = FALSE)  TrainSet <- data1[train,]  ValidSet <- data1[-train,]  summary(TrainSet)  summary(ValidSet) | |
| 1  2  3  4  5  6  7  8  9  10  11  12 | | > summary(TrainSet)   BuyingPrice Maintenance  NumDoors   NumPersons BootSpace    Safety    Condition   high :313   high :287   2    :305   2   :406   big  :416   high:396   acc  :264   low  :292   low  :317   3    :300   4   :399   med  :383   low :412   good : 52   med  :305   med  :303   4    :295   more:404   small:410   med :401   unacc:856   vhigh:299   vhigh:302   5more:309                                     vgood: 37  > summary(ValidSet)   BuyingPrice Maintenance  NumDoors   NumPersons BootSpace    Safety    Condition   high :119   high :145   2    :127   2   :170   big  :160   high:180   acc  :120   low  :140   low  :115   3    :132   4   :177   med  :193   low :164   good : 17   med  :127   med  :129   4    :137   more:172   small:166   med :175   unacc:354   vhigh:133   vhigh:130   5more:123                                     vgood: 28 | |

Now, we will create a Random Forest model with default parameters and then we will fine tune the model by changing ‘mtry’. We can tune the random forest model by changing the number of trees (ntree) and the number of variables randomly sampled at each stage (mtry). According to Random Forest package description:

Ntree: Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.

Mtry: Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (sqrt(p) where p is number of variables in x) and regression (p/3)

# Create a Random Forest model with default parameters

model1 <- randomForest(Condition ~ ., data = TrainSet, importance = TRUE)

model1

> model1

Call:

 randomForest(formula = Condition ~ ., data = TrainSet, importance = TRUE)

               Type of random forest: classification

                     Number of trees: 500

No. of variables tried at each split: 2

        OOB estimate of  error rate: 3.64%

Confusion matrix:

      acc good unacc vgood class.error

acc   253    7     4     0  0.04166667

good    3   44     1     4  0.15384615

unacc  18    1   837     0  0.02219626

vgood   6    0     0    31  0.16216216

By default, number of trees is 500 and number of variables tried at each split is 2 in this case. Error rate is 3.6%.

# Fine tuning parameters of Random Forest model

model2 <- randomForest(Condition ~ ., data = TrainSet, ntree = 500, mtry = 6, importance = TRUE)

model2

> model2

Call:

 randomForest(formula = Condition ~ ., data = TrainSet, ntree = 500,      mtry = 6, importance = TRUE)

               Type of random forest: classification

                     Number of trees: 500

No. of variables tried at each split: 6

        OOB estimate of  error rate: 2.32%

Confusion matrix:

      acc good unacc vgood class.error

acc   254    4     6     0  0.03787879

good    3   47     1     1  0.09615385

unacc  10    1   845     0  0.01285047

vgood   1    1     0    35  0.05405405

# Create a Random Forest model with default parameters

model1 <- randomForest(Condition ~ ., data = TrainSet, importance = TRUE)

model1

# Fine tuning parameters of Random Forest model

model2 <- randomForest(Condition ~ ., data = TrainSet, ntree = 500, mtry = 6, importance = TRUE)

model2

When we have increased the mtry to 6 from 2, error rate has reduced from 3.6% to 2.32%. We will now predict on the train dataset first and then predict on validation dataset.

# Predicting on train set

predTrain <- predict(model2, TrainSet, type = "class")

# Checking classification accuracy

table(predTrain, TrainSet$Condition)

> table(predTrain, TrainSet$Condition)

predTrain acc good unacc vgood

    acc   264    0     0     0

    good    0   52     0     0

    unacc   0    0   856     0

    vgood   0    0     0    37

# Predicting on Validation set

predValid <- predict(model2, ValidSet, type = "class")

# Checking classification accuracy

mean(predValid == ValidSet$Condition)

table(predValid,ValidSet$Condition)

> mean(predValid == ValidSet$Condition)

[1] 0.9884393

> table(predValid,ValidSet$Condition)

predValid acc good unacc vgood

    acc   117    0     2     0

    good    1   16     0     0

    unacc   1    0   352     0

    vgood   1    1     0    28

In case of prediction on train dataset, there is zero misclassification; however, in the case of validation dataset, 6 data points are misclassified and accuracy is 98.84%. We can also use function to check important variables. The below functions show the drop in mean accuracy for each of the variables.

# To check important variables

importance(model2)

varImpPlot(model2)